Python

Programmaufbau

```
#Bibliotheken einbinden
import turtle
import random

#Globale Variablen
name = "Manfred"

#Funktionsblock
def nameSetzen(nameNeu):
    global name
    name = nameNeu

#Hauptblock
nameSetzen("Gustav")
turtle.forward(random.randint(1, 10))
```

Operatoren

Vergleichsoperatoren

Operator	Bedeutung	Beispiel
==	ist gleich	a == b
!=	ist ungleich	a != b
>	größer	a > b
<	kleiner	a < b
>=	größer gleich	a >= b
⇐	kleiner gleich	a ⇐ b

Logische Operatoren

Operator	Bedeutung	Beispiel
and	und	(a == b) and (z > y)
or	oder	(a == b) or (z > y)
not	nicht	not (a == b)

Rechenoperatoren

Zur Durchführung von Berechnungen stehen verschiedene Rechenoperatoren zur Verfügung:

Operator	Zeichen	Beispiel	Ergebnis
Addition	+	3 + 4	7
Subtraktion	-	5 - 11	-6
Multiplikation	*	3 * 6	18
Division	/	9 / 2	4.5
Potenz	**	3 ** 4	81

```
zahl_a = zahl_x + zahl_y
zahl_b = zahl_x - 20
zahl_c = zahl_x * zahl_y
zahl_d = zahl_x / 5
zahl_e = zahl_x ** 4
```

Einseitige Auswahlstruktur

```
if bedingung:
   anweisung(en)
```

Beispiel

```
if x > y:
   turtle.forward(5)
```

Zweiseitige Auswahlstruktur

Die Alternative besteht aus einer Bedingung, deren Wahrheitswert überprüft wird. Je nach Ergebnis dieser Prüfung wird einer von zwei Anweisungsblöcken ausgeführt.

```
if bedingung:
   anweisung(en)
else:
   anweisung(en)
```

Beispiel

```
if x > y:
   turtle.forward(5)
else:
   turtle.forward(20)
```

Mehrseitige Auswahlstruktur

```
if bedingung:
   anweisung(en)
elif bedingung:
   anweisung(en)
else:
   anweisung(en)
```

Beispiel

```
if x > y:
   turtle.forward(5)
elif y > z:
   turtle.forward(10)
else:
   turtle.forward(15)
```

Zählschleife

```
for Variable in Sequenz:
   Anweisung1
   Anweisung2
   ...
   Anweisungn
```

Beispiel

```
for i in range(0,5):
   turtle.forward(100)
   turtle.left(90)
```

Zufallszahlen

https://docs.python.org/3/library/random.html

Eine natürliche Zufallszahl gibt uns folgende Anweisung zurück:

```
random.randint(m, n)
```

Der Aufruf `random.randint(1, 10)` liefert also eine natürliche Zahl zwischen 1 und 10, der Aufruf `random.randint(5, 8)` eine natürliche Zahl zwischen 5 und 8.

```
#Paket importieren
import random
#Anweisung
random.randint(ersteZahl, letzeZahl)
```

Beispiel

```
import random
zahl = random.randint(1, 100)
print(zahl)

turtle.forward(random.randint(1, 10))
```

Funktionen ohne Parameter

```
def bezeichner():
    Anweisungen
```

Beispiel

```
#Funktionsblock
#Hier werden Funktionen definiert, z.B.
def vor_und_zurueck():
   turtle.forward(100)
   turtle.right(180)
   turtle.forward(100)
   turtle.right(180)


######################
#Hauptblock
#Hier folgt der Aufruf der Funktionen, z.B.
vor_und_zurueck()
```

Funktionen mit Parameter

Das sind Funktionen, denen man Informationen übermittelt, die dann von der Funktion verarbeitet werden und bei jedem Aufruf ein darauf aufbauendes Ergebnis liefern. Die Informationen, die die Funktionen erhalten, nennt man **Parameter**.

Beispiel

```
#Funktion mit Parametern
def nutzer_gruessen(vorname, nachname):
    print("Herzlich willkommen,", vorname, nachname)

#Funktionsaufruf
nutzer_gruessen("Manfred", "Meyer")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#Eingabeaufforderung zur Veranschaulichung eines weiteren Beispiels
vornameEingabe = input("Bitte Vornamen eingeben: ")
nachnameEingabe = input("Bitte Nachname eingeben: ")

#Funktionsaufruf mit Variablen
nutzer_gruessen(vornameEingabe, nachnameEingabe)
```

Funktionen mit Rückgabewert

Programmierer können nicht nur Parameter an eine Funktion übergeben, sondern auch Ergebnisse mit einer Funktion an den Aufrufort zurückliefern. Diese Ergebnisse heißen **Rückgabewerte**. Rückgabewerte können sowohl Zahlen, Zeichenketten als auch Wahrheitswerte sein.

Beispiel

Als Beispiel hier ein Programm, welches das Quadrat einer eingegebenen Zahl errechnet und ausgibt:

```
def berechne_quadratzahl(zahl):
    quadratzahl = zahl * zahl
    return quadratzahl

zahl = float(input("Geben Sie bitte eine Zahl ein: "))
ergebnis = berechne_quadratzahl(zahl)
print("Die Quadratzahl von", zahl, "ist", ergebnis)
```

Turtle Grafik

Bibliothek `turtle graphics` - https://docs.python.org/3.3/library/turtle.html

Häufige Anweisungen

```
#turtle.forward(distanz)
turtle.forward(10)
```

```
#turtle.right(winkel)
turtle.right(90)

#turtle.left(winkel)
turtle.left(180)

#turtle.back(distanz)
turtle.back(10)

#turtle.goto(x,y)
turtle.goto(100,-100)

#Ausrichtung der Turtle bestimmen (0, 90, 180 oder 270)
turtle.setheading(0)

#turtle.dot(size=None, *color)
turtle.dot(20, "blue")
```

Bilddatei exportieren

```
import turtle
import time

timestr = time.strftime("%Y%m%d-%H%M%S")
ts = turtle.getscreen()

ts.getcanvas().postscript(file="dateiname" + timestr + ".eps")
```

Die .eps-Datei konvertieren (z.B. png oder jpg): https://www.epsconverter.com/de.html

Farbmodus bestimmen

```
s1 = turtle.Screen()
#rgb-Farben
s1.colormode(255)

#Stiftfarbe bestimmen
turtle.pencolor(255,255,255)
```

rgb-Farbtabelle: http://www.am.uni-duesseldorf.de/de/Links/Tools/farbtabelle.html

Kreisbogen zeichnen

```
import turtle
import random
```

```
import math

#Funktion Kreisbogen mit radius (r) und winkel (w)
def kreisbogen(r, w):
    for num in range (0,int(w/10)):
        turtle.left(2)
        turtle.forward((r*2*math.pi)/36)
        turtle.left(2)

kreisbogen(50,200)
```

Python im Unterrichtswiki

- Alternative
- Eingabe & Ausgabe
- Farben verwenden
- Funktionen mit Rückgabewert
- Funktionen ohne Parameter
- Python
- Python - Programmieraufgaben 1
- Spielwiese - Python Online Editor
- Turtle bewegen
- Variablen
- Verschachtelung
- Zählschleife